

# Tomography

Generated by Doxygen 1.8.7

Mon Aug 13 2018 11:36:10



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Namespace Documentation</b>	<b>5</b>
3.1	plot Namespace Reference . . . . .	5
3.1.1	Variable Documentation . . . . .	5
3.1.1.1	atext . . . . .	5
3.1.1.2	av_distances . . . . .	5
3.1.1.3	ax2 . . . . .	5
3.1.1.4	current . . . . .	5
3.1.1.5	data_file . . . . .	6
3.1.1.6	lineM . . . . .	6
3.1.1.7	lineN . . . . .	6
3.1.1.8	lineS . . . . .	6
3.1.1.9	lineT . . . . .	6
3.1.1.10	M . . . . .	6
3.1.1.11	N . . . . .	6
3.1.1.12	non_physical . . . . .	6
3.1.1.13	S . . . . .	6
3.1.1.14	total_time . . . . .	6
3.1.1.15	x_values . . . . .	6
<b>4</b>	<b>File Documentation</b>	<b>7</b>
4.1	enm-test.cpp File Reference . . . . .	7
4.1.1	Function Documentation . . . . .	7
4.1.1.1	main . . . . .	7
4.2	enm-test.h File Reference . . . . .	7
4.2.1	Macro Definition Documentation . . . . .	8
4.2.1.1	DEBUG_PRINT_DISTANCE_AVERAGES . . . . .	8

4.2.1.2	SHOW_PROGRESS	8
4.2.2	Typedef Documentation	8
4.2.2.1	MatrixXc	8
4.3	estimation.cpp File Reference	8
4.3.1	Function Documentation	8
4.3.1.1	d	8
4.3.1.2	enm_estimate_XYZ	8
4.3.1.3	linear_estimate_XYZ	8
4.4	estimation.h File Reference	8
4.4.1	Macro Definition Documentation	9
4.4.1.1	DEBUG_PRINT_ENM_OUTPUT	9
4.4.2	Typedef Documentation	9
4.4.2.1	MatrixXc	9
4.4.3	Function Documentation	9
4.4.3.1	enm_estimate_XYZ	9
4.4.3.2	linear_estimate_XYZ	9
4.5	plot.py File Reference	9
4.6	progress.cpp File Reference	10
4.6.1	Function Documentation	10
4.6.1.1	show_progress	10
4.7	progress.h File Reference	10
4.7.1	Function Documentation	10
4.7.1.1	show_progress	10
4.8	proj.cpp File Reference	10
4.8.1	Function Documentation	10
4.8.1.1	make_projector	10
4.9	proj.h File Reference	10
4.9.1	Typedef Documentation	11
4.9.1.1	MatrixXc	11
4.9.2	Function Documentation	11
4.9.2.1	make_projector	11
4.10	simulation.cpp File Reference	11
4.10.1	Function Documentation	11
4.10.1.1	random_density	11
4.10.1.2	random_unitary	11
4.10.1.3	simulate	11
4.11	simulation.h File Reference	11
4.11.1	Macro Definition Documentation	12
4.11.1.1	_USE_MATH_DEFINES	12
4.11.1.2	DEBUG_PRINT_MEASUREMENTS	12

4.11.1.3	DEBUG_PRINT_RANDOM_DENSITY	12
4.11.2	Typedef Documentation	12
4.11.2.1	MatrixXc	12
4.11.3	Function Documentation	12
4.11.3.1	random_density	12
4.11.3.2	random_unitary	12
4.11.3.3	simulate	13
4.12	stats.cpp File Reference	13
4.12.1	Function Documentation	13
4.12.1.1	distance_fid	13
4.12.1.2	distance_fid_2	13
4.12.1.3	distance_op	13
4.12.1.4	distance_trace	13
4.12.1.5	mean	13
4.13	stats.h File Reference	13
4.13.1	Macro Definition Documentation	14
4.13.1.1	DEBUG_PRINT_DISTANCES	14
4.13.2	Typedef Documentation	14
4.13.2.1	MatrixXc	14
4.13.3	Function Documentation	14
4.13.3.1	distance_fid	14
4.13.3.2	distance_fid_2	14
4.13.3.3	distance_op	14
4.13.3.4	distance_trace	14
4.13.3.5	mean	14
<b>Index</b>		<b>15</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">plot</a> . . . . .	5
--------------------------------	---





## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">enm-test.cpp</a>	7
<a href="#">enm-test.h</a>	7
<a href="#">estimation.cpp</a>	8
<a href="#">estimation.h</a>	8
<a href="#">plot.py</a>	9
<a href="#">progress.cpp</a>	10
<a href="#">progress.h</a>	10
<a href="#">proj.cpp</a>	10
<a href="#">proj.h</a>	10
<a href="#">simulation.cpp</a>	11
<a href="#">simulation.h</a>	11
<a href="#">stats.cpp</a>	13
<a href="#">stats.h</a>	13



## Chapter 3

# Namespace Documentation

### 3.1 plot Namespace Reference

#### Variables

- tuple `data_file` = `open(sys.argv[1], 'r')`
- tuple `av_distances` = `np.array([0,0,0])`
- list `non_physical` = `[]`
- list `x_values` = `[]`
- tuple `lineM` = `data_file.readline()`
- list `M` = `lineM[1]`
- tuple `lineN` = `data_file.readline()`
- list `N` = `lineN[1]`
- tuple `lineS` = `data_file.readline()`
- list `S` = `lineS[1]`
- tuple `current` = `line.split(',')`
- tuple `lineT` = `data_file.readline()`
- list `total_time` = `lineT[1]`
- tuple `ax2` = `ax1.twinx()`
- tuple `atext`

#### 3.1.1 Variable Documentation

##### 3.1.1.1 tuple `plot.atext`

#### Initial value:

```
1 = AnchoredText("Number of purity parameters: " + str(M) + "\n"
2               + "Density matrices: " + str(N) + "\n"
3               + "Samples per measurement: " + str(S) + "\n"
4               + "Total running time: " + str(total_time),
5               loc=2)
```

##### 3.1.1.2 tuple `plot.av_distances` = `np.array([0,0,0])`

##### 3.1.1.3 tuple `plot.ax2` = `ax1.twinx()`

##### 3.1.1.4 tuple `plot.current` = `line.split(',')`

3.1.1.5 tuple plot.data\_file = open(sys.argv[1], 'r')

3.1.1.6 tuple plot.lineM = data\_file.readline()

3.1.1.7 tuple plot.lineN = data\_file.readline()

3.1.1.8 tuple plot.lineS = data\_file.readline()

3.1.1.9 tuple plot.lineT = data\_file.readline()

3.1.1.10 list plot.M = lineM[1]

3.1.1.11 list plot.N = lineN[1]

3.1.1.12 list plot.non\_physical = []

3.1.1.13 list plot.S = lineS[1]

3.1.1.14 list plot.total\_time = lineT[1]

3.1.1.15 list plot.x\_values = []

# Chapter 4

## File Documentation

### 4.1 enm-test.cpp File Reference

```
#include "enm-test.h"
```

Include dependency graph for enm-test.cpp:

#### Functions

- int [main](#) ()

#### 4.1.1 Function Documentation

##### 4.1.1.1 int main ( )

### 4.2 enm-test.h File Reference

```
#include <iostream>
#include <complex>
#include <cstdlib>
#include <ctime>
#include <random>
#include <fstream>
#include <iomanip>
#include <chrono>
#include "Eigen/Dense"
#include "simulation.h"
#include "estimation.h"
#include "stats.h"
#include "proj.h"
#include "progress.h"
```

Include dependency graph for enm-test.h: This graph shows which files directly or indirectly include this file:

#### Macros

- #define [DEBUG\\_PRINT\\_DISTANCE\\_AVERAGES](#)
- #define [SHOW\\_PROGRESS](#)

## Typedefs

- typedef Eigen::Matrix  
     < std::complex< double >  
     , Eigen::Dynamic,  
     Eigen::Dynamic > [MatrixXc](#)

### 4.2.1 Macro Definition Documentation

4.2.1.1 `#define DEBUG_PRINT_DISTANCE_AVERAGES`

4.2.1.2 `#define SHOW_PROGRESS`

### 4.2.2 Typedef Documentation

4.2.2.1 `typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> MatrixXc`

## 4.3 estimation.cpp File Reference

```
#include "estimation.h"
```

Include dependency graph for estimation.cpp:

## Functions

- [MatrixXc linear\\_estimate\\_XYZ](#) (double X\_data[], double Y\_data[], double Z\_data[], int S)
- double [d](#) (const std::vector< double > &x, std::vector< double > &grad, void \*f\_data)
- [MatrixXc enm\\_estimate\\_XYZ](#) (double X\_data[], double Y\_data[], double Z\_data[], int S)

### 4.3.1 Function Documentation

4.3.1.1 `double d ( const std::vector< double > &x, std::vector< double > &grad, void * f_data )`

4.3.1.2 `MatrixXc enm_estimate_XYZ ( double X_data[], double Y_data[], double Z_data[], int S )`

4.3.1.3 `MatrixXc linear_estimate_XYZ ( double X_data[], double Y_data[], double Z_data[], int S )`

## 4.4 estimation.h File Reference

```
#include <iostream>
#include "stats.h"
#include "Eigen/Dense"
#include <vector>
#include "nlopt.hpp"
```

Include dependency graph for estimation.h: This graph shows which files directly or indirectly include this file:

## Macros

- `#define DEBUG\_PRINT\_ENM\_OUTPUT`

## Typedefs

- typedef Eigen::Matrix  
< std::complex< double >  
, Eigen::Dynamic,  
Eigen::Dynamic > [MatrixXc](#)

## Functions

- [MatrixXc linear\\_estimate\\_XYZ](#) (double X\_data[], double Y\_data[], double Z\_data[], int S)
- [MatrixXc enm\\_estimate\\_XYZ](#) (double X\_data[], double Y\_data[], double Z\_data[], int S)

### 4.4.1 Macro Definition Documentation

4.4.1.1 `#define DEBUG_PRINT_ENM_OUTPUT`

### 4.4.2 Typedef Documentation

4.4.2.1 `typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> MatrixXc`

### 4.4.3 Function Documentation

4.4.3.1 `MatrixXc enm\_estimate\_XYZ ( double X_data[], double Y_data[], double Z_data[], int S )`

4.4.3.2 `MatrixXc linear\_estimate\_XYZ ( double X_data[], double Y_data[], double Z_data[], int S )`

## 4.5 plot.py File Reference

### Namespaces

- [plot](#)

### Variables

- tuple [plot.data\\_file](#) = open(sys.argv[1], 'r')
- tuple [plot.av\\_distances](#) = np.array([0,0,0])
- list [plot.non\\_physical](#) = []
- list [plot.x\\_values](#) = []
- tuple [plot.lineM](#) = data\_file.readline()
- list [plot.M](#) = lineM[1]
- tuple [plot.lineN](#) = data\_file.readline()
- list [plot.N](#) = lineN[1]
- tuple [plot.lineS](#) = data\_file.readline()
- list [plot.S](#) = lineS[1]
- tuple [plot.current](#) = line.split(',')
- tuple [plot.lineT](#) = data\_file.readline()
- list [plot.total\\_time](#) = lineT[1]
- tuple [plot.ax2](#) = ax1.twinx()
- tuple [plot.atext](#)

## 4.6 progress.cpp File Reference

```
#include "progress.h"
```

Include dependency graph for progress.cpp:

### Functions

- int [show\\_progress](#) (std::chrono::time\_point< std::chrono::steady\_clock > start, double p)

### 4.6.1 Function Documentation

4.6.1.1 int show\_progress ( std::chrono::time\_point< std::chrono::steady\_clock > *start*, double *p* )

## 4.7 progress.h File Reference

```
#include <iostream>
```

```
#include <string>
```

```
#include <sstream>
```

```
#include <chrono>
```

Include dependency graph for progress.h: This graph shows which files directly or indirectly include this file:

### Functions

- int [show\\_progress](#) (std::chrono::time\_point< std::chrono::steady\_clock > start, double p)

### 4.7.1 Function Documentation

4.7.1.1 int show\_progress ( std::chrono::time\_point< std::chrono::steady\_clock > *start*, double *p* )

## 4.8 proj.cpp File Reference

```
#include "proj.h"
```

Include dependency graph for proj.cpp:

### Functions

- int [make\\_projector](#) (MatrixXc A, MatrixXc proj\_A[], double outcomes\_A[])

### 4.8.1 Function Documentation

4.8.1.1 int make\_projector ( MatrixXc A, MatrixXc *proj\_A*[], double *outcomes\_A*[] )

## 4.9 proj.h File Reference

```
#include <iostream>
```

```
#include "Eigen/Dense"
```

Include dependency graph for proj.h: This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef Eigen::Matrix  
     < std::complex< double >  
     , Eigen::Dynamic,  
     Eigen::Dynamic > [MatrixXc](#)

## Functions

- int [make\\_projector](#) ([MatrixXc](#) A, [MatrixXc](#) proj\_A[], double outcomes\_A[])

### 4.9.1 Typedef Documentation

4.9.1.1 typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> [MatrixXc](#)

### 4.9.2 Function Documentation

4.9.2.1 int [make\\_projector](#) ( [MatrixXc](#) A, [MatrixXc](#) proj\_A[], double *outcomes\_A*[] )

## 4.10 simulation.cpp File Reference

```
#include "simulation.h"
```

Include dependency graph for simulation.cpp:

## Functions

- [MatrixXc random\\_unitary](#) (std::mt19937 &generator)
- [MatrixXc random\\_density](#) (double x, std::mt19937 &generator)
- int [simulate](#) ([MatrixXc](#) dens, const [MatrixXc](#) proj[], const double meas[], int S, double sim\_dat[], std::mt19937 &generator)

### 4.10.1 Function Documentation

4.10.1.1 [MatrixXc random\\_density](#) ( double *x*, std::mt19937 & *generator* )

4.10.1.2 [MatrixXc random\\_unitary](#) ( std::mt19937 & *generator* )

Function: Generate random unitary

The method is to parametrise the unitary group and then select the right distribution for the parameters. See '2009 Ozols - How to generate a random unitary matrix', page 5, for more details.

4.10.1.3 int [simulate](#) ( [MatrixXc](#) *dens*, const [MatrixXc](#) *proj*[], const double *meas*[], int *S*, double *sim\_dat*[], std::mt19937 & *generator* )

## 4.11 simulation.h File Reference

```
#include <iostream>
```

```
#include <iomanip>
#include <cstdlib>
#include <ctime>
#include <chrono>
#include <random>
#include "Eigen/Dense"
#include <cmath>
```

Include dependency graph for simulation.h: This graph shows which files directly or indirectly include this file:

## Macros

- `#define` [DEBUG\\_PRINT\\_RANDOM\\_DENSITY](#)
- `#define` [DEBUG\\_PRINT\\_MEASUREMENTS](#)
- `#define` [\\_USE\\_MATH\\_DEFINES](#)

## Typedefs

- `typedef Eigen::Matrix`  
`< std::complex< double >`  
`, Eigen::Dynamic,`  
`Eigen::Dynamic > MatrixXc`

## Functions

- [MatrixXc random\\_unitary](#) (std::mt19937 &generator)
- [MatrixXc random\\_density](#) (double x, std::mt19937 &generator)
- int [simulate](#) ([MatrixXc](#) dens, const [MatrixXc](#) proj[], const double meas[], int S, double sim\_dat[], std::mt19937 &generator)

### 4.11.1 Macro Definition Documentation

4.11.1.1 `#define` [\\_USE\\_MATH\\_DEFINES](#)

4.11.1.2 `#define` [DEBUG\\_PRINT\\_MEASUREMENTS](#)

4.11.1.3 `#define` [DEBUG\\_PRINT\\_RANDOM\\_DENSITY](#)

### 4.11.2 Typedef Documentation

4.11.2.1 `typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> MatrixXc`

### 4.11.3 Function Documentation

4.11.3.1 [MatrixXc random\\_density](#) ( double *x*, std::mt19937 & *generator* )

4.11.3.2 [MatrixXc random\\_unitary](#) ( std::mt19937 & *generator* )

Function: Generate random unitary

The method is to parametrise the unitary group and then select the right distribution for the parameters. See '2009 Ozols - How to generate a random unitary matrix', page 5, for more details.

4.11.3.3 `int simulate ( MatrixXc dens, const MatrixXc proj[], const double meas[], int S, double sim_dat[], std::mt19937 & generator )`

## 4.12 stats.cpp File Reference

```
#include "stats.h"
```

Include dependency graph for stats.cpp:

### Functions

- double [distance\\_op](#) (MatrixXc A, MatrixXc B)
- double [distance\\_trace](#) (MatrixXc A, MatrixXc B)
- double [distance\\_fid](#) (const MatrixXc A, const MatrixXc B)
- double [distance\\_fid\\_2](#) (const MatrixXc A, const MatrixXc B)
- double [mean](#) (double array[], int N)

### 4.12.1 Function Documentation

4.12.1.1 `double distance_fid ( const MatrixXc A, const MatrixXc B )`

4.12.1.2 `double distance_fid_2 ( const MatrixXc A, const MatrixXc B )`

4.12.1.3 `double distance_op ( MatrixXc A, MatrixXc B )`

4.12.1.4 `double distance_trace ( MatrixXc A, MatrixXc B )`

4.12.1.5 `double mean ( double array[], int N )`

## 4.13 stats.h File Reference

```
#include "iostream"
#include "Eigen/Dense"
#include "Eigen/SVD"
```

Include dependency graph for stats.h: This graph shows which files directly or indirectly include this file:

### Macros

- `#define DEBUG\_PRINT\_DISTANCES`

### Typedefs

- `typedef Eigen::Matrix  
< std::complex< double >  
, Eigen::Dynamic,  
Eigen::Dynamic > MatrixXc`

### Functions

- double [distance\\_op](#) (MatrixXc A, MatrixXc B)
- double [distance\\_trace](#) (MatrixXc A, MatrixXc B)
- double [distance\\_fid](#) (const MatrixXc A, const MatrixXc B)
- double [distance\\_fid\\_2](#) (const MatrixXc A, const MatrixXc B)
- double [mean](#) (double array[], int N)

### 4.13.1 Macro Definition Documentation

4.13.1.1 `#define DEBUG_PRINT_DISTANCES`

### 4.13.2 Typedef Documentation

4.13.2.1 `typedef Eigen::Matrix<std::complex<double>, Eigen::Dynamic, Eigen::Dynamic> MatrixXc`

### 4.13.3 Function Documentation

4.13.3.1 `double distance_fid ( const MatrixXc A, const MatrixXc B )`

4.13.3.2 `double distance_fid_2 ( const MatrixXc A, const MatrixXc B )`

4.13.3.3 `double distance_op ( MatrixXc A, MatrixXc B )`

4.13.3.4 `double distance_trace ( MatrixXc A, MatrixXc B )`

4.13.3.5 `double mean ( double array[], int N )`

# Index

atext

plot, [5](#)

ax2

plot, [5](#)

current

plot, [5](#)

M

plot, [6](#)

N

plot, [6](#)

plot, [5](#)

atext, [5](#)

ax2, [5](#)

current, [5](#)

M, [6](#)

N, [6](#)

S, [6](#)

S

plot, [6](#)